AN564

Using the PWM

INTRODUCTION

The PIC16/17 family of RISC-like microcontrollers has been designed to provide advanced performance and a cost-effective solution for a variety of applications. This application report provides examples which illustrate the uses of Pulse Width Modulation (PWM) using the PIC17C42 Timer1 or Timer2 modules. These examples may be modified to suit the specific needs of your application.

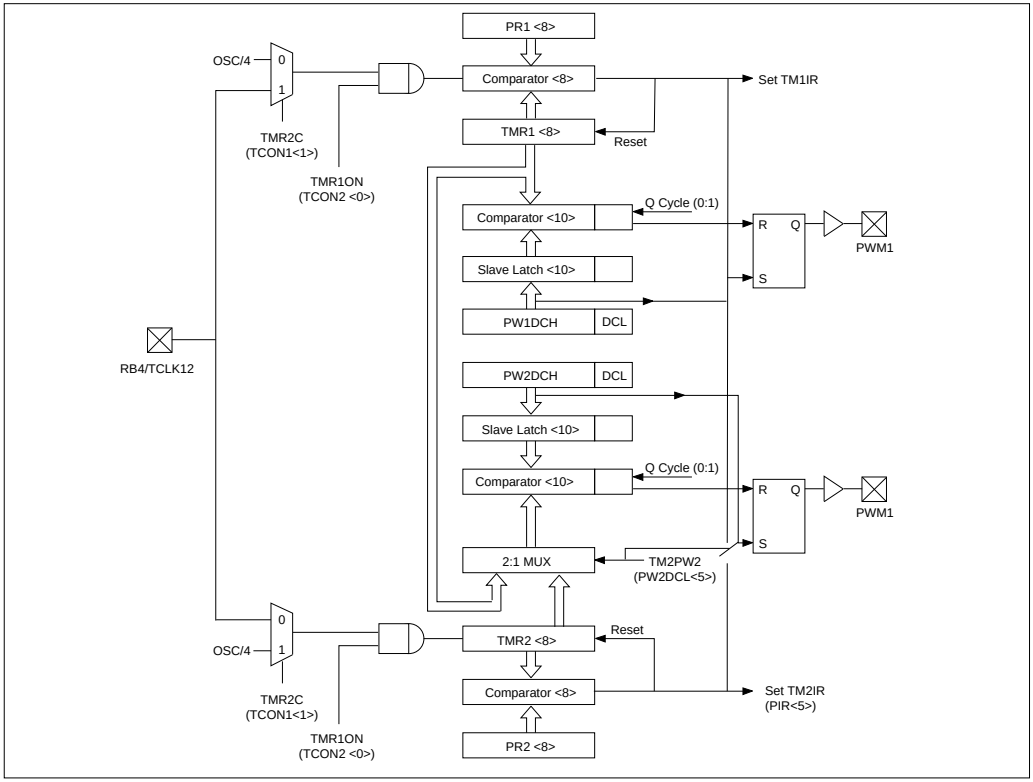
This Application Note describes the operation of the PWM. They include the following topics:

1. Simple PWM Operation
2. Variable Period / Variable Duty Cycle PWM
3 External Clock for Timer Timebase (ramifications/issues)

The listing file for the Variable Period / Variable Duty Cycle example can be found in Appendix A. The source files can be found on the Microchip BBS. On directions on how to access the Microchip BBS please refer to DS30128, which can also be found in the Microchip Embedded Control Handbook (Literature Number DS00092).

4

FIGURE 1 - TIMER1 AND TIMER2 BLOCK DIAGRAM WITH PWM MODE



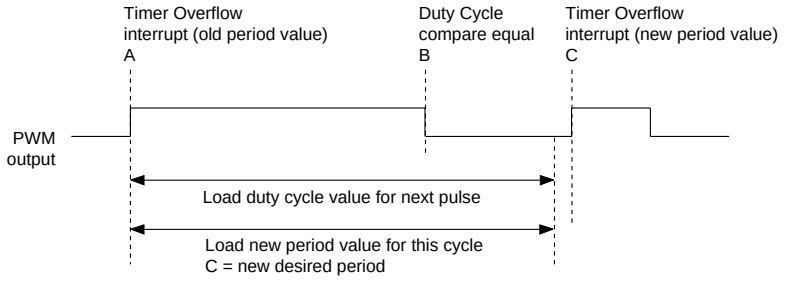
The control registers that are utilized by Timer1 and Timer2 are shown in Table 1. Shaded Boxes are control bits that are not used by the Timer1 nor Timer2 module.

TABLE 1 - REGISTERS ASSOCIATED WITH TIMER3 AND CAPTURE

Name	BANK	ADDR	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
INTSTA		0x06	PEIR	RTXIR	RTCIR	INTIR	PEIE	RTXIE	RTCIE	INTIE
CPUSTA		0x07	-	-	STKAV	GLINTD	$\overline{TO}$	$\overline{PD}$	-	-
TMR1	2	0x10	Timer1 Register							
TMR2	2	0x11	Timer2 Register							
PR1	2	0x14	Timer1 Period Register							
PR2	2	0x15	Timer2 Period Register							
PIR	1	0x16	IRB	TM3IR	TM2IR	TM1IR	CA2IR	CA1IR	TBMT	RBFL
PIE	1	0x17	IEB	TM3IE	TM2IE	TM1IE	CA2IE	CA1IE	TXIE	RCIE
PW1DCL	3	0x10	PWM1 Duty Cycle Low Register							
PW2DCL	3	0x11	PWM2 Duty Cycle Low Register							
PW1DCL	3	0x12	PWM1 Duty Cycle High Register							
PW2DCL	3	0x13	PWM2 Duty Cycle High Register							
TCON1	3	0x16	CA2ED1	CA2ED0	CA1ED1	CA1ED0	$\overline{16/8}$	TMR3C	TMR2C	TMR1C
TCON2	3	0x17	CA2OVF	CA1OVF	PWM2ON	PWM1ON	$\overline{CA1/PR3}$	TMR3ON	TMR2ON	TMR1ON

Care must be taken when loading values into the PWM registers. These registers are the duty cycle registers (PWxDCH:PWxDCL) and the period register (PRx). Figure 2 shows the proper update timing of these values.

FIGURE 2 - TIMING FOR UPDATING THE DUTY CYCLE REGISTERS AND PERIOD REGISTER



NOTE: When updating the period register, the value loaded must be greater than the timer value. If the period value is less then the timer value, the duty cycle value is not latched and the timer is not reset (to 00h) until the next Timer = Period match. This causes the current cycle to not output as expected and cause a “glitch”.

NOTE: It is generally good practice to load the new period value into the PRx register as soon as possible after Point A.

SIMPLE PWM OPERATION

Simple PWM operation is where the period of the PWM output remains constant, and only the duty cycle is modified. The PWM can operate in either of two modes:

- Hi-resolution mode-the PWxDCL register is modified
- Standard resolution mode - the PWxDCL register is not modified

When operating in the standard-resolution mode, only the PW-DCH register is ever modified. Since this takes only a single cycle, this can be done at any time. Also since the period is remaining constant this may be done without any PWM interrupt software overhead.

When operating in the high-resolution mode both the PWxDCH:PWxDCL register pair is modified. Since this is a multicycle update, care needs to be taken that the "new" PWM duty cycle value is not latched until the update is complete. If the duty cycle is latched before this update is complete, the duty cycle will display a “glitch”. If the PWxDCH is written first, the maximum error is 3 Q-cycles (187.5 ns @ 16 MHz). If the PWxDCL is written first, the maximum error is also 3 Q-cycles (187.5 ns @ 16 MHz), with the PWxDCH delayed by one PWM period. This may be acceptable for some applications. If this is not acceptable for your application then a subroutine can be written to ensure that these duty cycle writes are not done when the timer will equal the period. One implementation of this subroutine (PWM_UD) is used in the Variable Period / Variable Duty Cycle PWM example. This is discussed in the following section, with the listing in Appendix A.

Additional code examples can be found in application note AN539 in the Embedded Control Handbook.

VARIABLE PERIOD / VARIABLE DUTY CYCLE PWM

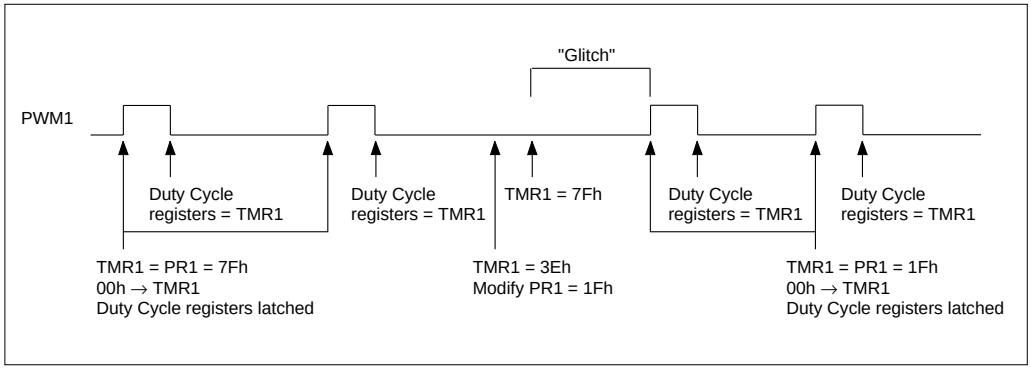
In a variable period / variable duty cycle PWM both the duty cycle of the PWM as well as the frequency (period) of the PWM are modified.

The PIC17C42’s hardware double buffers the duty cycle registers, but the period registers are not double buffered. What this means is that you can modify the duty cycle registers, but the value will only be latched when the timer register equals period register. Since the period register is not buffered, as the period register is modified this becomes the “new” period. This means that care must be taken when modifying the period register. The most common problem would be to modify the period register resulting in a “glitch” to occur. This “glitch” occurs when the period register is modified with a value that is less than the present timer value. The timer does not have a match with the old period value, and continues to count until the timer register equals period register.

Figure 3, shows an example where the period (PR1) register = 7Fh. Then the period is modified to a smaller value (PR1 = 1Fh) without checking that the value in Timer1 (TMR1) register = 3Eh. Since the new period (PR1) value is less then the present timer (TMR1) value, a glitch has occurred.

4

FIGURE 3 - MODIFYING PERIOD REGISTER “GLITCH”



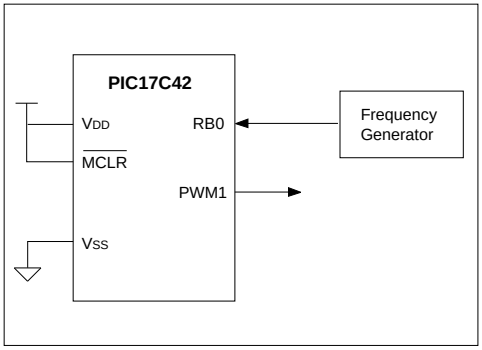
Care must be taken when writing a 10-bit duty cycle value. Since this requires two register writes, the Timer equals period could occur between these two writes, which would give a duty cycle that was not as expected. The cases are as follows:

1. If the duty cycle low (DCL) register is written, and then the Timer equals period. The old DCH register and the new DCL register becomes the duty cycle.
2. If the duty cycle high (DCH) register is written, and then the Timer equals period. The new DCH register and the old DCL register becomes the duty cycle

At the following occurrence of the timer equaling the period, the second register written would be updated. The subroutine PWM_UD (Appendix A) ensures that these duty cycle writes are not done when the timer will equal the period.

A software example of a variable period / variable duty cycle is shown in Appendix A. In this example the period is double buffered in software, and the new period value is loaded in the timer overflow interrupt service routine. When the new duty cycle needs to be loaded. The device connections are shown in Figure 4. This program has two PWM settings (period / duty cycle combinations) that are switched between depending on the level on pin RB0. A frequency generator was used to give a low frequency signal on the RB0 pin. Figure 5 shows an example of the input and output waveforms.

FIGURE 4 - APPLICATION HARDWARE SETUP

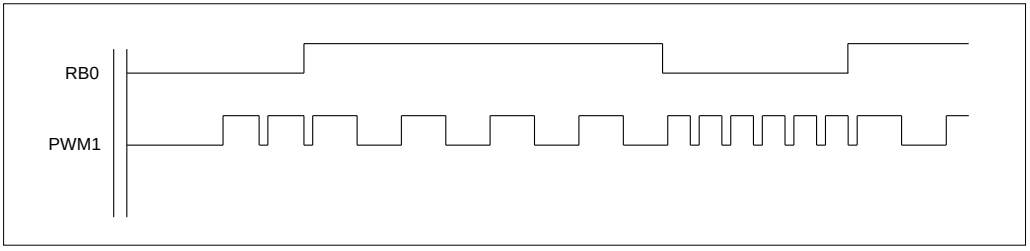


The program listing in Appendix A implements this example, Figure 8 is the flowchart of the program. This example may be modified to suit the particular needs of your application. The Table 3 is a summary of the requirements for this program (@ 16 MHz):

TABLE 3 - PROGRAM REQUIREMENTS

Code Size:	52 Words
RAM used:	11 Bytes
Interrupt Service Routine time	3.0 usec
Subroutine time	4.5 usec 6.0 usec
Maximum PWM frequency:	200 KHz
PWM Accuracy:	62.5 nsec

FIGURE 5 - EXAMPLE APPLICATION WAVEFORMS



EXTERNAL CLOCK FOR TIMER TIMEBASE

The counters used for the time base of the PWM outputs can be software selected to operate from an external clock source. This allows a lower frequency PWM to be achieved. Doing this brings up new issues that must be understood for the application.

One of these issues is clock synchronization. All external clocks must be synchronized to the internal operating speed of the microcontroller, as shown in figure 9. When this synchronization occurs the PWM output is not truly operating from the external clock, but actually the internal synchronized clock. This leads to a "jitter" of the output to the clock. This jitter is caused from the delta time between the external clock and the synchronized clock not being constant. The synchronization errors are:

Duty cycle error = +/- Tcy

Period error = +/- Tcy

If you needed to run the PWM at a low frequency, and also want to reduce the "jitter" from the use of an external asynchronous clock, a PWM output could be used as the synchronous clock source. When the clock is synchronized to the device the clock error is always constant, so there is no jitter. Figure 7 shows this example.

FIGURE 7 - PWM OUTPUT TO GENERATE A SYNCHRONOUS CLOCK

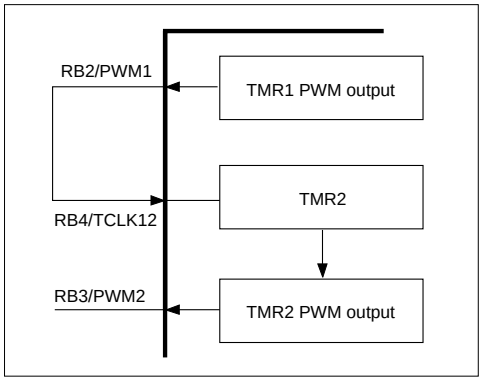
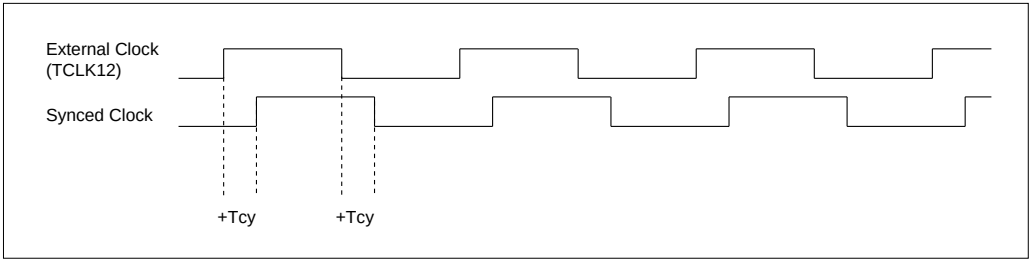


FIGURE 6 - EXTERNAL CLOCK SYNCHRONIZATION



Another use is where precise timing of updates need to be done, but not at the frequency of the PWM output. In this discussion, TMR1 is used as the time-base of a constant frequency PWM output. TMR1 uses the internal clock of the device and TMR2 uses the external clock input. TMR2 will get the clock input from the PWM2 output.

The PWM output is a constant frequency variable duty cycle output. The PW1DCH:PW1DCL register pair contain the variable duty cycle value of PWM1 output. The PW2DCH:PW2DCL register pair is set for a fixed duty cycle (50%) for the PWM2 output.

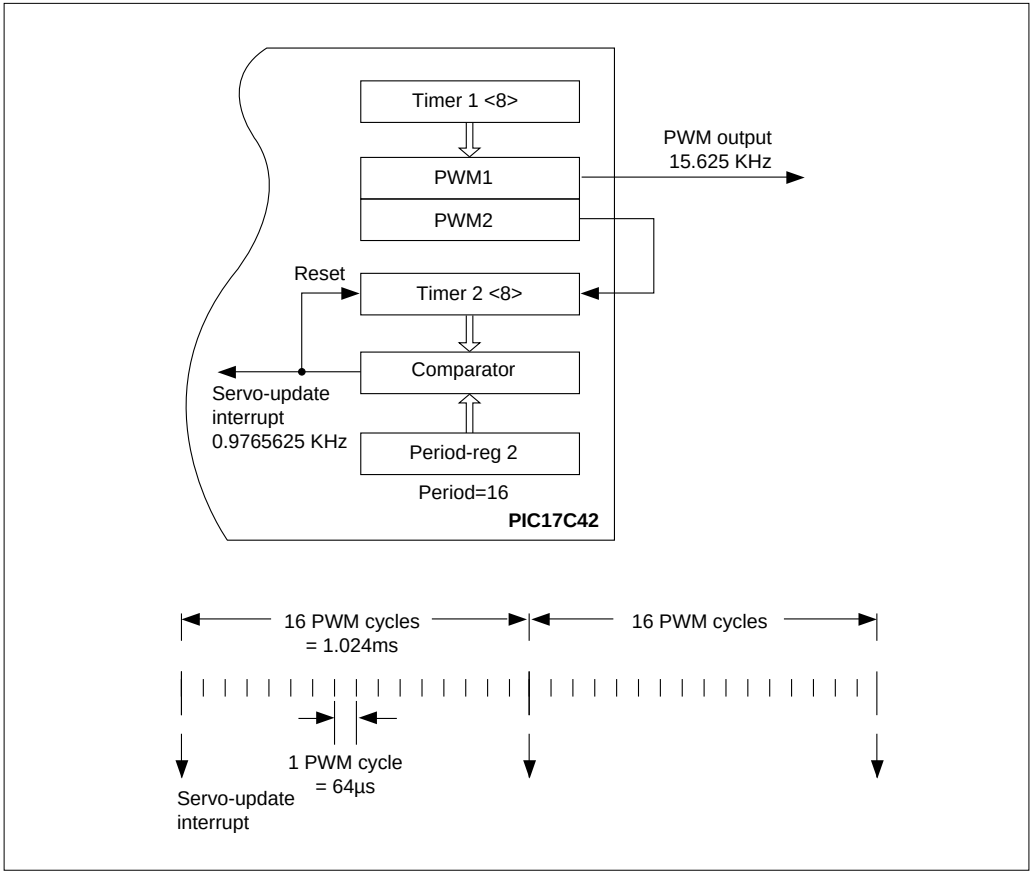
The PWM outputs could be programmed to have a frequency of 20 KHz, so to reduce audible noise. The PWM2 signal is connected to the RB4/TCLK12, as shown in Figure 8. The PR2 register could be loaded with 14h (20), to give an interrupt every 1 KHz. This interrupt can then trigger tasks, such as updating the duty cycle of PWM1. This is useful in motor control as well as other applications where the update rate is less then the PWM frequency.

CONCLUSION

The PIC17C42's PWM features offer a high performance solution at a lower system cost then previously available. The versatility of PWMs make the PIC17C42 ideal for motor control applications (ses AN532) and many industrial control applications.

Author: Mark Palmer
Logic Products Division

FIGURE 8 - SAMPLING SCHEME



APPENDIX A: LISTING FILE

```
MPASM B0.54                                                    PAGE  1


                      LIST   P=17C42, T=ON, C=120, N=0

                    ; This is the basic outline for a program that generates a
                    ; variable PWM output. The PWM's period and duty cycle can
                    ; be varied. The new period (NEW_PR1) and the new duty cycle
                    ; (NEW_DC1 and NEW_DC1Q) are loaded by the user program.
                    ; The peripheral interrupt routine loads the new period value
                    ; (frequency) into the PR1 register. A subroutine (PWM_UD)
                    ; is also used to ensure that the 10-bit duty cycle registers
                    ; are updated in the same PWM cycle, i.e. the timer match does not
                    ; occur between two duty cycle register writes.
                    ;
                    ; The duty cycle value gets latched on the overflow (Period match)
                    ; of the timer. The period value gets modified as soon as the period
                    ; register is changed. Therefore care must be taken in updating
                    ; the period register. In cases where the period value is modified
                    ; to a smaller value, we must ensure that the Timer counter is less
                    ; then this value when the period register is updated (TMR1 < new PR1).
                    ; If TMR1 is greater then PR1, the counter will count to FFh, rollover
                    ; to 00H, and only cause the overflow interrupt when it then reaches
                    ; the period value. This would give a wrong PWM output.
                    ;
                    ; In this example the event which cause the PWM to be updated
                    ; is an asynchronous event. A low frequency signal was placed on
                    ; port pin RB0.
                    ; For a high level the PWM registers are updated as follows:
                    ;    PR1 = 7Fh, PW1DCH = 3Fh, and PW1DCL = 40h
                    ; For a low level the PWM registers are updated as follows:
                    ;    PR1 = 1Fh, PW1DCH = 07h, and PW1DCL = 80h
                    ;
                    ; Do the EQUate table
                    ;
0020                NEW_DC1         EQU     0x20        ; New PWM1 duty cycle value
0021                NEW_DC1Q        EQU     0x21        ;
0022                NEW_PR1         EQU     0x22        ; New PWM1 period value
0025                PWM_WIN         EQU     0x25        ; Register for the PWM window cycle count
0026                CALC_PR         EQU     0x26        ; Calculated period value
0027                FLAG_REG        EQU     0x27        ; Register for flag bits
                    ;
001A                DC1H            EQU     0x1A        ; PWM registers for high time
001B                DC1QH           EQU     0x1B
001C                PR1H            EQU     0x1C
                    ;
001D                DC1L            EQU     0x1D        ; PWM registers for low time
001E                DC1QL           EQU     0x1E
001F                PR1L            EQU     0x1F
                    ;
                    ;
07FF                END_OF_PROG_MEM EQU 0x07FF
                    ;
0004                ALUSTA          EQU     0x04
0006                CPUSTA          EQU     0x06
0007                INTSTA          EQU     0x07
000A                W               EQU     0x0A
                    ;
0011                DDRB            EQU     0x11        ; Bank 0
0012                PORTB           EQU     0x12
                    ;
0016                PIR             EQU     0x16        ; Bank 1
0017                PIE             EQU     0x17
                    ;
```



```
0010                TMR1            EQU     0x10            ; Bank 2
0011                TMR2            EQU     0x11
0012                TMR3l           EQU     0x12
0013                TMR3h           EQU     0x13
0014                PR1             EQU     0x14
0015                PR2             EQU     0x15
0016                PR3L            EQU     0x16
0017                PR3h            EQU     0x17
                    ;
0010                PW1DCL          EQU     0x10            ; Bank 3
0011                PW2DCL          EQU     0x11
0012                PW1DCH          EQU     0x12
0013                PW2DCH          EQU     0x13
0016                TCON1           EQU     0x16
0017                TCON2           EQU     0x17

                                    ORG     0x0000          ; Origin for the RESET vector
0000 C02B                           GOTO    START           ; On reset, go to the start of
                                                            ;   the program
                                    ORG     0x0008          ; Origin for the external RA0/INT
                                                            ;   interrupt vector
0008 C07C                           GOTO    EXT_INT         ; Goto the ext. interrupt
                                                            ;   on RA0/INT routine
                                    ORG     0x0010          ; Origin for the RTCC
                                                            ;   overflow interrupt vector
0010 C07D                           GOTO    RTCCINT         ; Goto the RTCC overflow interrupt
                                                            ;   routine
                                    ORG     0x0018          ; Origin for the external
                                                            ;   RA1/RT interrupt vector
0018 C07E                           GOTO    RT_INT          ; Goto the ext. interrupt on
                                                            ;   RA1/RT routine
                                    ORG     0x0020          ; Origin for the interrupt vector
                                                            ;   of any enabled peripheral
                    ;
                    ; The interrupt routine for any peripheral interrupt, This routine
                    ; only deals with Timer1 interrupt.
                    ;
                    ; Time required to execute interrupt routine. Not including
                    ; interrupt latency (time to enter into the interrupt routine)
                    ;
                    ;       case1 - only T1 overflow            = 12 cycles
                    ;       case2 - Other                       = Infinite Loop
                    ;
                    ;
0020 B801           PER_INT         MOVLB   1               ; Select register Bank 1
0021 9416                           BTFSS   PIR,4           ; Did Timer1 overflow?
0022 C022           ERROR           GOTO    ERROR           ; Not a Timer1 overflow.
                                          ; No other interrupts should
                                          ; be enabled, so error.
                    ;
                    ; Once the enabled Timer1 overflow occurs, the period register
                    ; is loaded. This PWM waveform will remain until the PWM duty
                    ; cycle and / or period is updated. Until such update, there is no
                    ; S/W overhead from T1 interrupts (T1 interrupts can be disabled).
                    ;
                    ; NOTE: If PW1DCH >= PR1, then the duty cycle of this PWM output
                    ;       is 100%.
                    ;
                    ; NOTE: The new Period register (PR1) value, must always be greater
                    ;       than the value in the Timer1 register (TMR1). If a PR1 value
                    ;       is loaded that is less then the TMR1 value, the timer will
                    ;       continue to count until it reaches the PR1 value. I.E. TMR1
                    ;       will overflow at FFh and the count to the new PR1 value.
                    ;       Minimum PR1 value is 0Ah, due to time to load new values and
                    ;       execute the peripheral interrupt service routine.
                    ;
0023 8C16           T1OVFL          BCF     PIR,4           ; Clear Overflow interrupt flag
0024 B802                           MOVLB   2               ; Bank2
0025 7422                           MOVFP   NEW_PR1,PR1     ; Load this period value
```

```
0026 B801                           MOVLB   1                   ; Bank 0
0027 8C17                           BCF     PIE, 4              ; Disable T1 interrupt
                                                                ;   (until transition on PORTB0)
0028 B800                           MOVLB   0                   ; Bank 0
0029 3F12                           BTG     PORTB, 7            ; Transition PortB 7 pin (H->L,
002A 0005                           RETFIE                      ; Return from Interrupt

                    ;
                    ; This is the start of the program.
                    ;
002B 8406           START           BSF     CPUSTA,4            ; Disable ALL interrupts via the
                                                                ;   Global Interrupt Disable
                                                                ;   (GLINTD) bit.
                                                                ;
                    MAIN                                        ; Place Main program here
002C B803                           MOVLB   3                   ; Select register Bank 3
002D 2817                           CLRF    TCON2,0             ; Stop the timers, Single Capture
002E B070                           MOVLW   0x070               ; Initalize TCON1 so that
002F 0116                           MOVWF   TCON1               ;   T1 (8-bit), T2 (8-bit),
                                                                ;   and T3 run off the internal
                                                                ;   system clock. Timer3 uses
                                                                ;   period register
0030 B00D                           MOVLW   0x0D                ; Load the PWM window cycle value
0031 0125                           MOVWF   PWM_WIN             ;
                    ;
0032 B800                           MOVLB   0                   ; Select register Bank 0
0033 2B11                           SETF    DDRB, 1             ; Port B is an input
0034 2912                           CLRF    PORTB, 1            ; Set output values to 0 (for
0035 8F11                           BCF     DDRB, 7             ; PORTB7 is an output. Used to
0036 2927                           CLRF    FLAG_REG, 1         ; Clear the Flag registers
                    ;
                    ; Load registers with the PWM values that we will switch between. One set
                    ; for the time PORTB0 is high and another set for when low.
                    ;
                    ; For a high level the PWM registers are updated as follows:
                    ;    PR1 = 7Fh, PW1DCH = 3Fh, and PW1DCL = 40h
                    ;    At 16Mhz this gives a period of 31.75 us, and a duty cycle of 16.625 us
                    ; For a low level the PWM registers are updated as follows:
                    ;    PR1 = 1Fh, PW1DCH = 07h, and PW1DCL = 80h
                    ;    At 16Mhz this gives a period of 7.75 us, and a duty cycle of 6.00 us
                    ;
0037 B803                           MOVLB   3                   ; Bank 3
0038 B03F                           MOVLW   0x3F                ; The Duty Cycle initial value is
0039 4A1A                           MOVPF   W, DC1H             ;   50% of the initial period
003A B040                           MOVLW   0x40                ;
003B 4A1B                           MOVPF   W, DC1QH            ; Duty Cycle low = 01
003C B007                           MOVLW   0x07                ; The Duty Cycle initial value is
003D 4A1D                           MOVPF   W, DC1L             ;   25% of the initial period
003E B080                           MOVLW   0x80                ;
003F 4A1E                           MOVPF   W, DC1QL            ; Duty Cycle low = 10
                    ;
0040 B802                           MOVLB   2                   ; Bank 2
0041 B07F                           MOVLW   0x7F                ;
0042 4A1C                           MOVPF   W, PR1H             ; The initial period value is 50%
                                            ;   of full scale (for High)
0043 B01F                           MOVLW   0x1F
0044 4A1F                           MOVPF   W, PR1L             ; The initial period value is
                                                                ;   of full scale (for Low)
                    ;
                    ;
                    ; Default PWM values should be set, and the timer should be started
                    ; and the interrupts enabled.
                    ;
0045 B0F0                           MOVLW   0xF0                ; Load the Period register
0046 0114                           MOVWF   PR1                 ;
0047 B803                           MOVLB   3                   ; Select register Bank 3
0048 B0C0                           MOVLW   0xC0                ; Load the T1 duty cycle register
0049 0112                           MOVWF   PW1DCH              ;
004A 0110                           MOVWF   PW1DCL              ; effectively loaded with 0
```

```
004B B031                       MOVLW   0x31            ;** Enable PWM1 and PWM2 outputs
004C 0117                       MOVWF   TCON2           ;**   and turn on Timer1.
004D 8307                       BSF     INTSTA,3        ; Turn on Peripheral Interrupts
004E B801                       MOVLB   1               ; Select register Bank 1
004F B010                       MOVLW   0x10            ; Enable Timer1 overflow
0050 0117                       MOVWF   PIE             ;   Interrupts (when GLINTD = 0)
0051 8C06                       BCF     CPUSTA,4        ; Enable ALL interrupts
0052 B800                       MOVLB   0               ; Bank 0

                        ;
                        ; Only need to update PWM values on the first occurance of a new level on
                        ; Else loop waiting for level to change.
                        ;
0053 8827               HIGH1ST BCF     FLAG_REG, 0     ; First time in loop (this cycle) = True
0054 9012               HIGHCYC BTFSS   PORTB,  0       ; Is PortB0 low
0055 C05F                       GOTO    LOW1ST          ; PORTB0 = L
0056 9827                       BTFSC   FLAG_REG, 0     ; Is this the First High time(this cycle)?
0057 C054                       GOTO    HIGHCYC         ; Loop looking for low signal on PortB0
0058 8027                       BSF     FLAG_REG, 0     ; Set First time in loop(this cycle)=False
                        ;
                        ; Here is where we update the PWM values (period and Duty cycle) for high
                        ;
0059 B803                       MOVLB   3               ; Bank 3
005A 5A20                       MOVPF   DC1H, NEW_DC1   ;
005B 5B21                       MOVPF   DC1QH, NEW_DC1Q ;
005C 5C22                       MOVPF   PR1H, NEW_PR1   ;
005D E06B                       CALL    PWM1_UD         ;
005E C054                       GOTO    HIGHCYC         ; Loop looking for low signal on
                        ;
                        ;
005F 8827               LOW1ST  BCF     FLAG_REG, 0     ; First time in loop (this
0060 9812               LOWCYC  BTFSC   PORTB, 0         ; Is PortB0 high
0061 C053                       GOTO    HIGH1ST         ; PORTB0 = H
0062 9827                       BTFSC   FLAG_REG, 0     ; Is this the First Low time
0063 C060                       GOTO    LOWCYC          ; Loop looking for high signal on PortB0
0064 8027                       BSF     FLAG_REG, 0     ; First time in loop (this    ;
                        ; Here is where we update the PWM values (period and Duty cycle) for low
                        ;
0065 B803                       MOVLB   3               ; Bank 3
0066 5D20                       MOVPF   DC1L, NEW_DC1   ;
0067 5E21                       MOVPF   DC1QL, NEW_DC1Q ;
0068 5F22                       MOVPF   PR1L, NEW_PR1   ;
0069 E06B                       CALL    PWM1_UD         ;
006A C060                       GOTO    LOWCYC          ; Loop looking for high signal


                        ;
                        ; This code segment ensure that all PWM values (period and duty cycle)
                        ; are updated at the same time. This is done by ensuring that the Timer
                        ; is at least PWM_WIN (0Dh) cycles before the PR1 value (PR1 - PWM_WIN >
                        ; If not a “glitch” could occur in the PWM waveform. When only the 1st duty
                        ; cycle register is latched for this PWM cycle, and the following PWM period
                        ; will latch the 2nd duty cycle register.
                        ;
006B 8406               PWM1_UD BSF     CPUSTA, 4       ; Disable Global Interrupts
006C B802                       MOVLB   2               ; Bank 2
006D 6A10                       MOVFP   TMR1, W         ; Load W reg. with Timer1 value
006E 0414                       SUBWF   PR1, 0          ; PR1 - TMR1 -> W reg.
006F 3025                       CPFSLT  PWM_WIN         ; Check if Timer1 is about to
0070 C06B                       GOTO    PWM1_UD         ; Overflow would have occurred
                                                        ;   PWM updates, Delay a few
0071 B803                       MOVLB   3               ; Bank 3
0072 6A20                       MOVFP   NEW_DC1, W      ; Your New PWM MSB
0073 0112                       MOVWF   PW1DCH          ; Loaded in duty cycle buffer
0074 6A21                       MOVFP   NEW_DC1Q, W     ; Your New PWM LSB
0075 0110                       MOVWF   PW1DCL          ; Loaded in duty cycle buffer
0076 B801                       MOVLB   1               ; Back to Bank 1
0077 8C16                       BCF     PIR, 4          ; Clear T1 Overflow interrupt
```

```
0078 8417                           BSF     PIE, 4              ; Enable T1 int
0079 8C06                           BCF     CPUSTA, 4           ; Enable Global Interrupts
007A B800                           MOVLB   0                   ; Bank 0
007B 0002                           RETURN                      ;** this does not need to be
                                                                ;**  as a subroutine.
                    ;
                    ; Other Interrupt routines. (Not utilized in this example)
                    ;
007C 0005           EXT_INT         RETFIE                      ; RA0/INT interrupt routine
                                                                ;   (NOT used in this program)
007D 0005           RTCCINT         RETFIE                      ; RTCC overflow interrupt routine
                                                                ;   (NOT used in this program)
007E 0005           RT_INT          RETFIE                      ; RA1/RT interrupt routine
                                                                ;   (NOT used in this program)
                                                                ;
007F C02B           SRESET          GOTO    START               ; If program became lost, goto
                                                                ;   START and reinitalize.
                    ;
                    ;
                    ; When the executed address is NOT in the program range, the
                    ; 16-bit address should contain all 1's (a CALL 0x1FFF). At
                    ; this location you could branch to a routine to recover or
                    ; shut down from the invalid program execution.
                    ;
                                    ORG     END_OF_PROG_MEM     ;
07FF C07F                           GOTO    SRESET              ; The program has lost it's mind,
                                                                ; do a system reset
                                    END


Errors   :    0
Warnings :    0
```

NOTES:

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602 786-7200 Fax: 602 786-7277
Technical Support: 602 786-7627
Web: http://www.mchip.com/microhip

Atlanta
Microchip Technology Inc.
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770 640-0034 Fax: 770 640-0307

Boston
Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508 480-9990 Fax: 508 480-8575

Chicago
Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 708 285-0071 Fax: 708 285-0075

Dallas
Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 214 991-7177 Fax: 214 991-8588

Dayton
Microchip Technology Inc.
35 Rockridge Road
Englewood, OH 45322
Tel: 513 832-2543 Fax: 513 832-2841

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 455
Irvine, CA 92715
Tel: 714 263-1888 Fax: 714 263-1338

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 416
Hauppauge, NY 11788
Tel: 516 273-5305 Fax: 516 273-5335

AMERICAS (continued)

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408 436-7950 Fax: 408 436-7955

ASIA/PACIFIC

Hong Kong
Microchip Technology
Unit No. 3002-3004, Tower 1
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T. Hong Kong
Tel: 852 2 401 1200 Fax: 852 2 401 3431

Korea
Microchip Technology
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku,
Seoul, Korea
Tel: 82 2 554 7200 Fax: 82 2 558 5934

Singapore
Microchip Technology
200 Middle Road
#10-03 Prime Centre
Singapore 188980
Tel: 65 334 8870 Fax: 65 334 8850

Taiwan
Microchip Technology
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886 2 717 7175 Fax: 886 2 545 0139

EUROPE

United Kingdom
Arizona Microchip Technology Ltd.
Unit 6, The Courtyard
Meadow Bank, Furlong Road
Bourne End, Buckinghamshire SL8 5AJ
Tel: 44 0 1628 851077 Fax: 44 0 1628 850259

France
Arizona Microchip Technology SARL
2 Rue du Buisson aux Fraises
91300 Massy - France
Tel: 33 1 69 53 63 20 Fax: 33 1 69 30 90 79

Germany
Arizona Microchip Technology GmbH
Gustav-Heinemann-Ring 125
D-81739 Muenchen, Germany
Tel: 49 89 627 144 0 Fax: 49 89 627 144 44

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Pegaso Ingresso No. 2
Via Paracelso 23, 20041
Agrate Brianza (MI) Italy
Tel: 39 039 689 9939 Fax: 39 039 689 9883

JAPAN

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shin Yokohama
Kohoku-Ku, Yokohama
Kanagawa 222 Japan
Tel: 81 45 471 6166 Fax: 81 45 471 6122

9/22/95